CS787: Advanced Algorithms	**Scribe:** Derek Paulsen
Lecture 20: k-Server Problem Analysis; Convex Body Chasing	**Date:** 4/7/22

20.1 k-server problem, continued

Last time, we discussed the k-server problem,

- You have k-servers that are in a metric space (e.g. a line, 2-d euclidean space)
- Requests arrive online each at some point in the space
- Each request must be addressed by sending a single server to the request
- Minimize the total distance the servers move to service the requests

The naive greedy algorithm of sending the closest server to the request to this problem is not competitive. In this lecture we discuss the line metric, analyze the double coverage algorithm, and show that it is k-competitive. Recall the double coverage algorithm,

20.1.1 Double Coverage Algorithm

Algorithm 1 Double Coverage Algorithm

```
x ← the current positions of the servers
r ← the next request
if r ∉ [min(x), max(x)] then
    move the closest server to r
else
    i ← the closest server to the right of r
    j ← the closest server to the left of r
    d ← min{|x_i − r|, |x_j − r|}
    move servers x_i and x_j d units toward r
end if
```

Theorem 20.1.1 *The Double Coverage Algorithm is k-competitive*

Let σ be a sequence of requests, $\text{DC}(\sigma)$ be the cost the double coverage algorithm pays to service the requests and $\text{OPT}(\sigma)$ be the cost that OPT pays to service the requests.

Proof : For any sequence of requests σ we want to show,

$$\text{DC}(\sigma) \leq k\text{OPT}(\sigma) + C \tag{20.1.1}$$

Ideally, we would show that for each request, DC pays at most kOPT to service the request. However this is unfortunately not true (imagine if OPT already has a server on the request). Instead we use

a potential argument and show that for each request i, the change in $\mathrm{DC} = \Delta\mathrm{DC}$ plus the change in the potential, $\phi^i - \phi^{i-1} = \Delta\phi^i$ is less than k times the change in $\mathrm{OPT} = k\Delta\mathrm{OPT}$, that is,

$$\Delta\mathrm{DC}^i(\sigma) + \Delta\phi^i \leq k\Delta\mathrm{OPT}^i(\sigma) \tag{20.1.2}$$

Where ϕ is the potential function. Specifically, for request i if x^i is the position of our servers and y^i is the position of OPT's servers

$$\phi^i = \phi(x^i, y^i) = \phi_1^i + \phi_2^i = kM(x^i, y^i) + \sum_{h<l} d(x_h^i, x_l^i)$$

Where $\phi_1 = kM(x, y)$ is k times the minimum distance matching between x and y, and $\phi_2 = \sum_{h<l} d(x_h, x_l)$ is the pairwise distances all our servers x.

Note that if we sum 20.1.2 for each step the potentials cancel out and we are left with,

$$\mathrm{DC}(\sigma) + \phi^{last} - \phi^{first} \leq k\mathrm{OPT}(\sigma) + C$$

Since ϕ is bounded we get,

$$\mathrm{DC}(\sigma) \leq k\mathrm{OPT}(\sigma) + C$$

For ease of exposition we break each request into two stages, in stage one OPT moves, and stage two DC moves. Now consider the stage 1, where OPT moves distance d to service the request, consider the changes of each

- $\Delta\mathrm{OPT} = d$, since OPT moved d
- $\Delta\mathrm{DC} = 0$, since we didn't move yet
- $\Delta\phi_1 \leq kd$, since in the worse case, the cost of $M(x, y)$ increased by d
- $\Delta\phi_2 = 0$, since we didn't move yet

From this we get,

$$\begin{aligned} \Delta\mathrm{DC}^i(\sigma) + \Delta\phi^i &\leq k\Delta\mathrm{OPT}^i(\sigma) \\ 0 + kd + 0 &\leq kd \end{aligned}$$

Hence for stage one,

$$\Delta\mathrm{DC}^i(\sigma) + \Delta\phi^i \leq k\Delta\mathrm{OPT}^i(\sigma)$$

For stage two we have two cases,

1. The request is outside the servers, in which case, we send one server
2. The request is between two servers, in which case, we send two servers

For case one we have, suppose we move one server d' to service the request,

- $\Delta\text{OPT} = 0$, since OPT didn't move,
- $\Delta\text{DC} = d'$, since we moved a single server d
- $\Delta\phi_1 = -kd'$, since we moved to the same point as one OPT's servers to service the request
- $\Delta\phi_2 = (k-1)d'$, since we moved one server away from all the others

Summing these terms we get,

$$\begin{aligned} \Delta\text{DC}^i(\sigma) + \Delta\phi^i &\leq k\Delta\text{OPT}^i(\sigma) \\ d' - kd' + (k-1)d' &\leq 0 \\ 0 &\leq 0 \end{aligned}$$

For case two suppose again the we move both servers d',

- $\Delta\text{OPT} = 0$, since OPT didn't move,
- $\Delta\text{DC} = 2d'$, since we moved two servers d'
- $\Delta\phi_1 \leq 0$, since we moved to the same point as one OPT's servers to service the request, but then moved another server possible away from the one it was matched to
- $\Delta\phi_2 = -2d'$, since two servers moved closer to each other and the distances to between the servers that moved and didn't move cancel out

Summing these terms we get,

$$\begin{aligned} \Delta\text{DC}^i(\sigma) + \Delta\phi^i &\leq k\Delta\text{OPT}^i(\sigma) \\ 2d' - 2d' + 0 &\leq 0 \\ 0 &\leq 0 \end{aligned}$$

Therefore, we have shown that at each step,

$$\Delta\text{DC}^i(\sigma) + \Delta\phi^i \leq k\Delta\text{OPT}^i(\sigma)$$

Which implies,

$$\text{DC}(\sigma) \leq k\text{OPT}(\sigma) + C$$

Which by definition means that DC is k-competitive. $\square$

20.2 Convex Body Chasing

We now look at a problem related to the k-server problem called convex body chasing which is,

- you have a single server
- requests arrive online and are convex bodies
- service each request by move the server into the convex body
- minimize the distance your server travels for the sequence of requests

For a long time whether there was any competitive algorithm was an open problem. The first result to demonstrate a competitive algorithm was for chasing half-spaces (i.e. each convex body was a half-space) where the authors showed an algorithm that was 2^d-competitive, where d is the dimension of the space.

Proof : no algorithm is better than $\sqrt{d}$ competitive

Consider the sequence of requests, first is a halfspace, the second is a point on the boundary of the first halfspace. Clearly, the optimal solution is to just go to the second point immediately (since it satisfies both requests). However for any algorithm, no matter which point we go to first, the second request can be placed somewhere else the boundary of the halfspace, meaning the algorithm takes an L shaped path, which is at best $\sqrt{d}$-competitive with the direct (optimal) path.

More recently in 2019 it was shown that this problem can always be solved with competitive ratio d by solving the special case where each request is nested inside the previous one. Creating an algorithm for this special case can then be adapted to the general case. The algorithm to show this is simply

- Move to the Steiner Point of the request

Definition 20.2.1 (Steiner Point) *Each vertex of a convex body has a cone where it is the minimum point inside of the convex body if you are moving in a direction within the cone. The Steiner Point is the average of the vertexes of the convex body, weighted by the size of it's cone.*

Mathmatically, for a convex body k, the Steiner point $St(k)$ is,

$$St(k) = \int_{||\theta||=1} \nabla_k(\theta) d\theta, \quad \nabla_k(\theta) = argmax_{x \in k}\{x \text{ is the minimum point moving in direction } \theta\}$$

Where $||\theta|| = 1$ is all vectors on the unit sphere. This can be rewritten as,

$$St(k) = d \int_{||\theta||=1} S_k(\theta)\theta d\theta, \quad S_k(\theta) = max_{x \in k}\{x \text{ is the minimum point moving in direction } \theta\}$$

Claim : For nested convex bodies, moving to $St(k)$ is d-competitive.

Proof :

We begin with the sum of the cost for each step, for a sequence of requests $\sigma = \{k_1, ..., k_T\}$,

$$
\begin{aligned}
ALG(\sigma) &= \sum_{i=1}^{T} ||St(k_i) - St(k_{i-1})|| \quad \text{the cost of moving from each steiner point to the other} \\
&= \sum_{i=1}^{T} ||d \int_{||\theta||=1} (S_{k_i}(\theta) - S_{k_{i-1}}(\theta))\theta d\theta|| \quad \text{replace with definition from above} \\
&= d \int_{||\theta||=1} \sum_{i=1}^{T} ||S_{k_i}(\theta) - S_{k_{i-1}}(\theta)||\theta d\theta \quad \text{move sum inside} \\
&= d \int_{||\theta||=1} \sum_{i=1}^{T} ||S_{k_i}(\theta) - S_{k_{i-1}}(\theta)||d\theta \quad ||\theta|| = 1 \\
&= d \int_{||\theta||=1} ||S_{k_T}(\theta) - S_{k_1}(\theta)||d\theta \quad \text{the sum telescopes, leaving on the first and last term}
\end{aligned}
$$

It can then be shown that $\int_{||\theta||=1} S_{k_T}(\theta) - S_{k_1}(\theta) \leq \text{OPT} + C$, hence the algorithm is d-competitive.